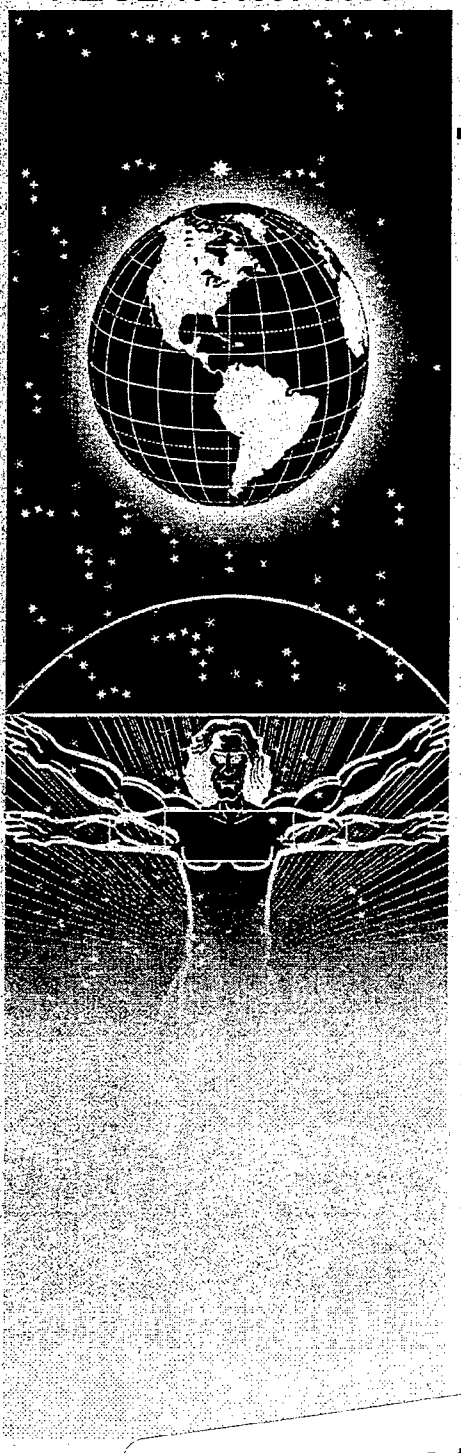
AL/OE-TR-1997-0063



UNITED STATES AIR FORCE
ARMSTRONG LABORATORY

Laser Atmospheric Attenuation Tables for LTAS

Brian J. Lund

TASC
4241 Woodcock Drive, Suite B-100
San Antonio, TX 78228

May 1997

Approved for public release;

distribution is unlimited.

Occupational and Environmental Health Directorate
Optical Radiation Division
8111 18th Street
Brooks AFB, TX 78235-5215

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

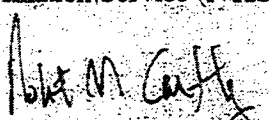
The mention of trade names or commercial products in this publication is for illustration purposes and does not constitute endorsement or recommendation for use by the United States Air Force.

The Office of Public Affairs has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

Government agencies and their contractors registered with Defense Technical Information Center (DTIC) should direct requests for copies to: Defense Technical Information Center, 8725 John J. Kingman Rd, Ste 0944, Ft. Belvoir, VA 22060-6218.

Non-government agencies may purchase copies of this report from: National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield VA 22161-2103.



ROBERT M. CARTLEDGE, Lt Col, USAF, BSC
Chief, Optical Radiation Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 1997	Final - February 1996 - June 1996

4. TITLE AND SUBTITLE

Laser Atmospheric Attenuation Tables for LTAS

5. FUNDING NUMBERS

C - F33615-92-C-0017
PE - 62202F
PR - 7757
TA - 02
WU - 99

6. AUTHOR(S)

Brian J. Lund

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

TASC
4241 Woodcock Drive, Ste B-100
San Antonio, TX 78228

8. PERFORMING ORGANIZATION

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Armstrong Laboratory
Occupational and Environmental Health Directorate
Optical Radiation Division
8111 18th Street
Brooks Air Force Base TX 78235-5215

10. SPONSORING/MONITORING

AL/OE-TR-1997-0063

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A description of the use of the program FASCODE to create tables of atmospheric attenuation coefficients and to incorporate them into the Laser Threat Analysis System (LTAS) is included in this report. FASCODE covers an incredible amount of atmospheric and light situations to compute the atmospheric attenuation coefficient. Incorporating FASCODE into LTAS would have been an enormous undertaking, and it would have slowed down the execution of LTAS considerably. The alternate approach that was taken included creating several tables of atmospheric attenuation coefficient only for the cases of interest to LTAS. These cases included nine laser wavelength lines, four atmospheric region models, four aerosol models, and five visibility ranges. The format of the tables is also described here as a reference, to facilitate adding more cases of interest for LTAS in the future.

14. SUBJECT TERMS
Aerosol models; Atmosphere; Atmospheric attenuation; Atmospheric models; Laser

15. NUMBER OF PAGES
72

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

This report describes the use of the program FASCODE to create tables of atmospheric attenuation coefficients to incorporate atmospheric attenuation of laser light in the Laser Threat Analysis System (LTAS). Tables of attenuation coefficients have been assembled for a number of laser wavelengths using several of the standard atmosphere and aerosol models incorporated into FASCODE (see the User Instructions for FASCOD3[1]). An understanding of terms such as "transmittance" is assumed. Previous studies have described the physics of laser light transmission through the atmosphere.[2-4]

The tables have been created for the geometry illustrated in Figure 1. A ground-based laser is assumed to be illuminating an approaching target aircraft, which is flying at an altitude h above the ground. The slant range S_r is the distance from the laser to the target aircraft.

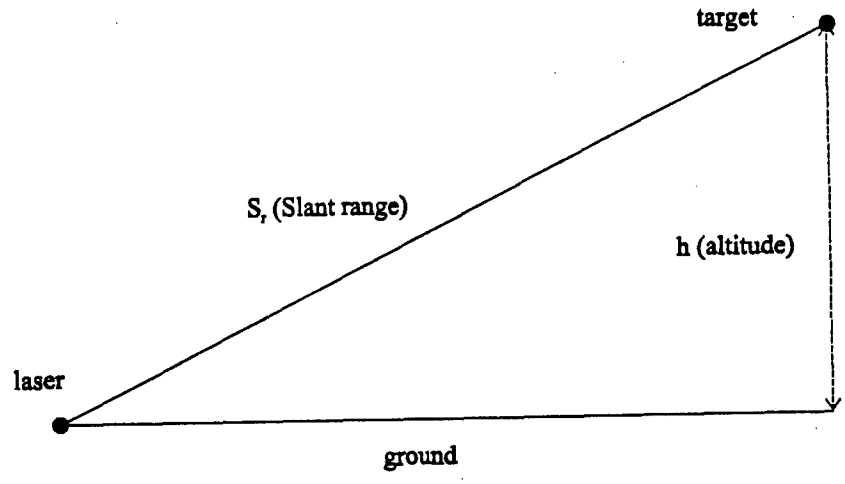


Figure 1. Geometry for atmospheric transmittance calculations.

The transmittance T of the laser light from the ground to an altitude h may be parameterized as an exponential function of the slant range S_r:

$$T(S_r) = \exp(-\mu(h) \cdot S_r) \qquad (1)$$

where $\mu(h)$ is the attenuation coefficient for the laser light transmitted from ground level to altitude h. The parameter μ depends on the wavelength λ of the laser light as well as the atmospheric conditions.

EXPONENTIAL PARAMETERIZATION

The validity of the exponential parameterization is demonstrated in Figure 2. This figure shows the transmittance as a function of slant range for λ = 1.064 μm from the ground to an altitude h = 500 ft for four different aerosol models. The symbols represent the results of FASCODE calculations of the transmittance. The attenuation coefficients μ were determined by fitting the points below S_r = 10 km with an exponential function of the form

$$T(S_r) = T_0 \exp(-\mu \cdot S_r). \qquad (2)$$

In general, it was found that $|T_0 - 1| < 10^{-3}$. It is therefore reasonable to set T_0 = 1.0 for the purposes of calculating the transmittance $T(S_r)$, which reduces Equation 2 to the form of Equation 1. (This will also simplify the form of the atmospheric attenuation tables, as only the values of μ need be tabulated.)

The curves in Figure 2 were calculated using Equation 1 and the values of μ shown on the figure. Note that the points at S_r = 20 km and 30 km were not included in the fits to determine μ, but are still well-represented by the exponential curves.

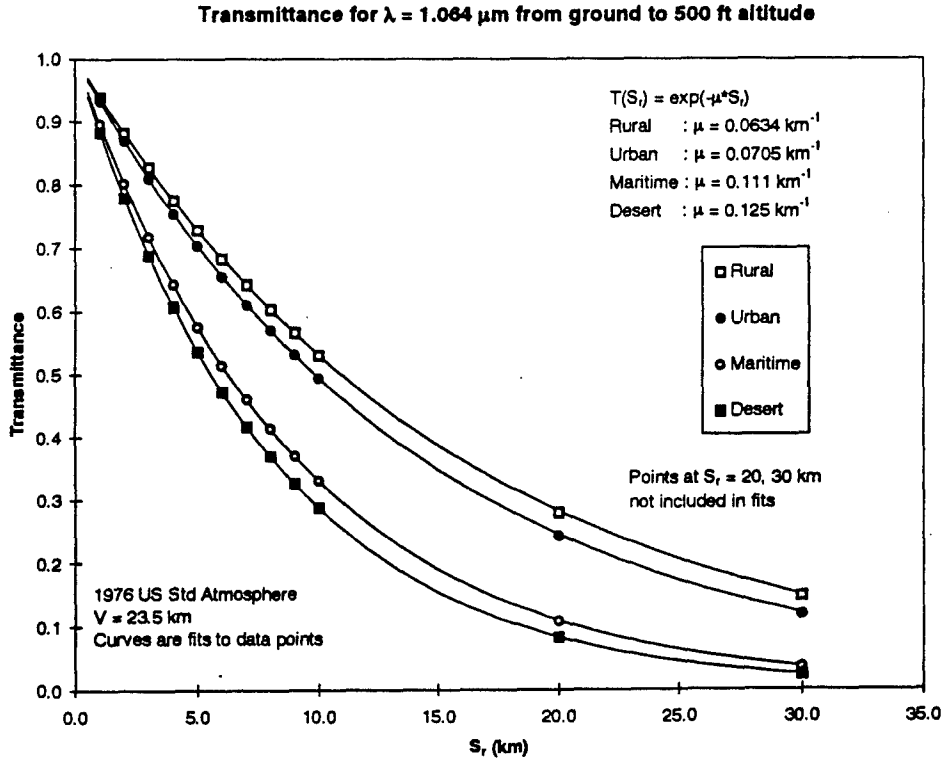


Figure 2. Comparison of exponential parameterization with FASCODE calculations of atmospheric transmittance.

A similar examination of the exponential parameterization has been performed for other wavelengths and altitudes (e.g., λ = 10.6 μm, h = 15,000 ft). Results similar to those shown in Figure 2 were found to hold in all cases.

ATMOSPHERIC MODELS

Table 1 lists the atmosphere models, aerosol models, and ground level visibility conditions for which attenuation coefficient tables have been calculated. These are from the list of standard models incorporated into FASCODE.

Table 1. List of Atmosphere Models, Aerosol Models, and Visibility Conditions used for Calculating Atmospheric Attenuation Coefficients.

Atmosphere Models	Aerosol Models	Visibility Ranges	(Desert Wind Speeds)
1976 US Standard	Rural	5.0 km (Haze)	25.68 m/s
Tropical	Urban	8.0 km (Med. Haze)	23.45 m/s
Midlatitude Summer	Maritime	15.0 km (Light Haze)	20.42 m/s
Midlatitude Winter	Desert	23.5 km (Clear)	17.21 m/s
		40.0 km (Very Clear)	12.97 m/s

Column 3 of this table includes labels for the various visibility ranges. The numerical values listed for the visibility V should be considered as the maximum range within that category. For example, when 15.0 km < V < 23.5 the atmospheric conditions are described as Clear.

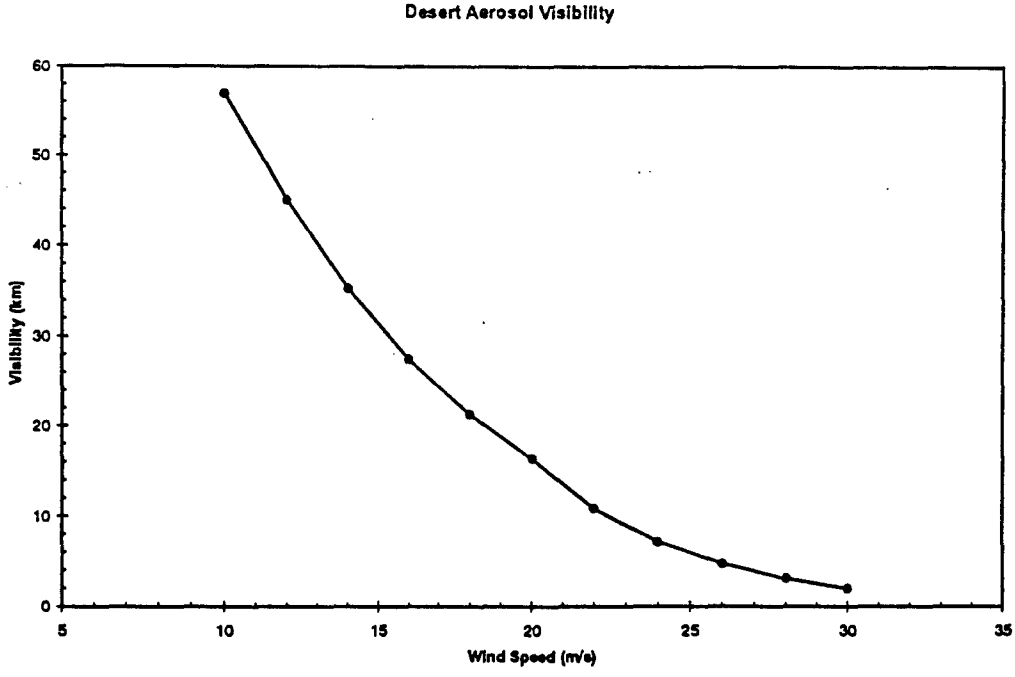


Figure 3. Visibility as a function of wind speed in the Desert aerosol model.

In the Desert aerosol model, the visibility range is determined by the wind speed, as illustrated in Figure 3. In calculating the Desert attenuation coefficient tables, the wind speed listed in column 4 of Table 1 was used. These wind speeds produce the values for the visibility listed in column 3.

FORMAT AND USE OF THE TABLES

For a given laser wavelength, the attenuation coefficients are tabulated in a series of data files, one file for each atmosphere-aerosol model combination. There are 16 such files for each wavelength (4 atmosphere models x 4 aerosol models). Within each file, the attenuation coefficients are listed in a series of columns, one column for each of the five visibility ranges listed in Table 1.

Figure 4 is an example of one of the files. This is the table of attenuation coefficients for $\lambda = 10.6$ μm passing through the Tropical atmosphere plus Rural aerosol. Each file has the same form. The first line of the file identifies the atmosphere and aerosol models, as well as the laser wavelength. The attenuation coefficients have been determined for altitudes h = 1000 ft to h = 20,000 ft, in 1000-ft increments. The coefficients are listed for each of the five visibility conditions shown in Table 1. Note that the attenuation coefficients μ are given in units of km^{-1}. When using Equation 1 the slant range S_r must therefore be in units of km.

Tropical, Rural, 10.6 um

Alt. (ft)	Attenuation Coefficients (km-1) V=5.0km	V=8.0km	V=15.0km	V=23.5km	V=40.0km
1000.0	3.88e-01	3.65e-01	3.47e-01	3.39e-01	3.34e-01
2000.0	3.60e-01	3.37e-01	3.18e-01	3.10e-01	3.05e-01
3000.0	3.34e-01	3.11e-01	2.92e-01	2.83e-01	2.79e-01
4000.0	3.09e-01	2.87e-01	2.68e-01	2.60e-01	2.56e-01
5000.0	2.83e-01	2.65e-01	2.48e-01	2.41e-01	2.37e-01
6000.0	2.60e-01	2.44e-01	2.30e-01	2.24e-01	2.20e-01
7000.0	2.40e-01	2.26e-01	2.14e-01	2.08e-01	2.05e-01
8000.0	2.20e-01	2.08e-01	1.97e-01	1.92e-01	1.89e-01
9000.0	2.01e-01	1.90e-01	1.81e-01	1.76e-01	1.73e-01
10000.0	1.84e-01	1.75e-01	1.66e-01	1.62e-01	1.59e-01
11000.0	1.70e-01	1.61e-01	1.53e-01	1.50e-01	1.47e-01
12000.0	1.57e-01	1.49e-01	1.42e-01	1.39e-01	1.36e-01
13000.0	1.46e-01	1.38e-01	1.32e-01	1.29e-01	1.27e-01
14000.0	1.36e-01	1.29e-01	1.23e-01	1.20e-01	1.18e-01
15000.0	1.27e-01	1.21e-01	1.15e-01	1.13e-01	1.11e-01
16000.0	1.20e-01	1.14e-01	1.09e-01	1.06e-01	1.04e-01
17000.0	1.13e-01	1.08e-01	1.03e-01	1.00e-01	9.86e-02
18000.0	1.07e-01	1.02e-01	9.71e-02	9.49e-02	9.33e-02
19000.0	1.02e-01	9.66e-02	9.21e-02	9.01e-02	8.86e-02
20000.0	9.67e-02	9.19e-02	8.77e-02	8.57e-02	8.43e-02

Figure 4. Table of atmospheric attenuation coefficients for $\lambda = 10.6$ μm passing through the Tropical atmosphere plus Rural aerosol models.

To use this table, one should take the value of μ listed for the nearest altitude greater than the altitude h of the approaching target aircraft. For example, assume an aircraft is approaching a CO_2 laser (wavelength λ = 10.6 μm) on a clear day (15 km < V < 23.5 km). The aircraft is approaching at an altitude of h = 9500 ft. The attenuation coefficient μ should be taken from the column labeled 'V=23.5km' for the altitude h = 10,000 ft. The value of the attenuation coefficient that should be used for this situation is $\mu = 1.62 \times 10^{-1}$ km^{-1}.

DESCRIPTION OF SYSTEM USED TO ASSEMBLE TABLES

Several C++ programs and DOS batch files were developed to help automate the process of making the tables of attenuation coefficients. Program, batch file, and sample input file listings are included in the appendixes of this report. The system was developed specifically to use the standard FASCODE atmosphere and aerosol models. To allow the use of a user-supplied atmosphere, some of the programs will need to be modified.

The PC version of FASCODE (FASCD3P.EXE) was used for all of the atmospheric transmittance calculations. This program requires two input files, called TAPE3 and TAPE5. The file TAPE3 contains molecular line-by-line absorption data extracted from the HITRAN92 data base in the region of the wavenumber ($k = 1/\lambda$) of interest. The file TAPE5 controls the FASCODE calculations, specifying the atmospheric conditions, path geometry, etc.

Below is a brief description of all the programs used to make the tables. While several of the programs listed below could be combined into one larger program, it was useful to keep their functions separated in order to be able to examine each step of the process. Note that all of these programs are designed to run in DOS.

LNFL92.EXE[5]

LNFL92.EXE[5] is the program used to extract molecular absorption data from the HITRAN92 CD. The execution of this program is controlled by an input file called TAPE5 (not the same as the FASCODE input file) which contains the wavenumber range and number of molecules for which data from HITRAN92 should be extracted. According to the LNFL92 instructions, molecular absorption data should be extracted for wavenumbers at least 25 cm^{-1} above and below the frequency range of interest. For the laser wavelength calculations (i.e., monochromatic), this range was extended to $\pm$50 cm^{-1} about the wavenumber ($k = 1/\lambda$) of the laser.

There are 31 molecules included in the HITRAN92 data base (see the User Instructions for FASCOD3[1] for a list). Although for most cases only a few molecules will dominate the absorption, a template input file was set up to extract data for all 31 molecules. This sped up the process of extracting data, because the input file TAPE5 only had to be edited to change the wavenumbers, without having to worry about how many molecules were needed for accurate calculations.

The extracted data are written by LNFL92 to the file TAPE3, which is immediately usable by FASCODE. In practice, these files were renamed and stored for future use.

FASCD3P.EXE[1]

FASCD3P.EXE[1] is the PC version of FASCODE. The execution of this program is controlled by the file TAPE5. A template for this input file, TEMPLATE.TP5, is included in the appendixes. This file is designed for calculations using the standard FASCODE atmosphere and aerosol models.

The template file is set up to calculate the optical depth τ (related to transmittance through $T = \exp(-\tau)$) for a series of slant paths from ground to a given altitude h. The resulting values of T are then fit with an exponential function of the slant range S_r (Equation 2) in order to find the attenuation coefficient for this altitude.

FASCD3P reads the molecular absorption data from the file TAPE3. It is important to make sure this file includes the wavenumbers of interest for the current calculation.

The output of FASCD3P is written to the file TAPE6.

FASREAD.EXE

FASREAD.EXE extracts slant range, optical depth, and altitude from the FASCD3P output file TAPE6. The optical depth is converted to transmittance, $T = \exp(-\tau)$, for the output of FASREAD.

EFIT.EXE

EFIT.EXE performs the exponential fit of T versus S_r. This program is designed specifically to have the output of FASREAD piped into it.

MUTAB.EXE

MUTAB.EXE takes the fit results from EFIT for a series of altitudes and organizes them into a nice, easy-to-read format. As used, the output from MUTAB contains all the fits for one atmosphere, aerosol, wavelength, and visibility combination.

FASIN.EXE

FASIN.EXE allows several of the parameters of the FASCODE input file TAPE5 to be easily altered via switches on the command line. The file TAPE5 must be of the standard form of TEMPLATE.TP5 (although the length may vary). The output of FASIN is written to TAPE5.NEW, which must be copied or renamed to TAPE5 before using with FASCD3P.

TRANSMIT.BAT

This batch file controls the execution of the programs necessary to calculate the attenuation coefficients for altitudes h from 1000 ft to 20,000 ft for one wavelength, atmosphere model, aerosol model, and visibility combination. It calls FASIN to set the parameters of TAPE5, then executes FASCD3P. FASREAD and EFIT are then called to produce the exponential fits of T versus S_r. Finally, MUTAB is called to organize all the fits.

TRANSDES.BAT

TRANSDES.BAT is similar to TRANSMIT.BAT, but it is tailored for use with the Desert aerosol. A separate batch file was necessary because the Desert aerosol visibility is determined by wind speed.

MAKETABL.EXE

MAKETABL.EXE assembles the five output files from MUTAB (called by TRANSMIT and TRANSDES) for a given wavelength, atmosphere, and aerosol combination into the final table form illustrated in Figure 4. The execution of this program is controlled by an input file, an example of which is included in the appendixes. Four standard input files have been created, one for each of the four atmosphere models: MAKEUS.INF, MAKETR.INF, MAKEMS.INF, and MAKEMW.INF.

GOUS.BAT, GOTR.BAT, GOMS.BAT, GOMW.BAT

These batch files run all the programs necessary to make the attenuation tables for a given wavelength for all four aerosol models using the 1976 US Standard, Tropical, Midlatitude Summer, and Midlatitude Winter atmosphere models, respectively. These batch files call MAKETABL using the .INF files listed above. They also delete the output files from EFIT and MUTAB, which are no longer needed once MAKETABL has been run to assemble the final tables.

GOALL.BAT

This batch file calls each of the GO files listed above. Once the system is set up properly (proper TAPE3 file, GO batch file, and .INF files edited for the wavelength of interest), all one needs to do to make the 16 attenuation tables for the given wavelength is to invoke GOALL and wait (patiently). Time for execution depends on the speed of the computer's processor and the size of the molecular absorption data file TAPE3. For $\lambda < 1$ μm, the TAPE3 files are about 12 kb in size. On a 166 MHz Pentium processor, the run time for GOALL is about 1-1/2 hours. For $\lambda = 10.6$ μm, however, TAPE3 is 1.7 Mb, and the run time increases to nearly one day. For this wavelength, a smaller input file TAPE5 was used, after checking to make sure the resulting values for the attenuation coefficients were not strongly affected.

INSTRUCTIONS FOR MAKING THE STANDARD TABLES

There are essentially four steps in making the standard 16 tables for a given wavelength. These steps mostly boil down to using an editor's find-and-replace command to change the wavelength in the batch and MAKETABL .INF files. This greatly reduces the possibility of making an error. Considering the length of time needed for the calculations, one doesn't want to have to start again from the beginning.

Note once again that this system is designed to use the standard models incorporated into FASCODE. If a user-supplied atmosphere is to be used, some modifications to this system are necessary.

Step 1

Make sure the file TAPE3 contains the molecular absorption data for the wavelength of interest. About the only indication that this has not been done will be that the FASCODE calculations seem to be going along awfully quick.

Step 2

Edit GOUS.BAT, GOTR.BAT, GOMS.BAT, and GOMW.BAT to change the wavelength (on the lines calling TRANSMIT.BAT and TRANSDES.BAT) to the value of interest.

Step 3

Edit MAKEUS.INF, MAKETR.INF, MAKEMS.INF, and MAKEMW.INF to change the wavelength values (in the label lines and the names of the MUTAB output files) to the value of interest.

Step 4

Start GOALL.BAT and wait!

REFERENCES

1. *User Instructions for FASCOD3.* (These instructions are obtainable from the Phillips Laboratory's Geophysics Division ftp site: 146.153.100.3, in the directory /pub/chet.)
2. Wolfe, W.L., and Zissis, G.J. (Eds.). (1978). *The Infrared Handbook.* Office of Naval Research, Department of the Navy.
3. Kneizys, F.X., Clough, S.A., Shettle, E.P., Rothman, L.S., and Fenn, R.W. (1978). Linear absorption and scattering of laser beams. AFGL-TR-84-0265.
4. Smith, H.J.P., Dube, D.J., Gardner, M.E., Clough, S.A., Kneizys, F.X., and Rothman, L.S. (1978). FASCODE - fast atmospheric signature code (spectral transmittance and radiance). AFGL-TR-78-0081.
5. *Instructions for Line File Creation for FASCOD3 - LNFL.* (These instructions are obtainable from Geophysics Division of Phillips Laboratory ftp site [see reference 1].)

APPENDIX A: FILE NAMING CONVENTIONS

APPENDIX A: FILE NAMING CONVENTIONS

This appendix describes the conventions used to name various files created during the process of assembling the tables of atmospheric attenuation coefficients. There are several files created to store intermediate results, as well as the final tables.

EFIT and MUTAB Output Files

The filenames for the output of these programs are supplied within the GOXX.BAT files, on the lines calling TRANSMIT.BAT and TRANSDES.BAT. The output of EFIT is called FNAME.DAT, and the output of MUTAB is called FNAME.TAB, where FNAME has the form FNAME = [m][a][v]. As indicated in Table A-1, [m] is a two-letter abbreviation of the atmosphere model, [a] is a three-letter abbreviation of the aerosol model, and [v] is an integer indicating the visibility *V*.

Table A-1. Naming Convention for EFIT and MUTAB Output Files.

FNAME = [m][a][v]		
[m] (atmosphere)	[a] (aerosol)	[v] (visibility)
us = 1976 US Standard	rur = Rural	5 = 5.0 km
tr = Tropical	urb = Urban	8 = 8.0 km
ms = Midlatitude Summer	mar = Maritime	15 = 15.0 km
mw = Midlatitude Winter	des = Desert	235 = 23.5 km
		40 = 40.0 km

Example: The file USDES15.TAB is the output of MUTAB for the 1976 US Standard atmosphere, Desert aerosol, V = 15.0 km, for the wavelength of current interest.

The GOXX.BAT files delete all the FNAME.DAT and FNAME.TAB files after they have been used by MAKETABL. In order to save these files for examination, the delete commands must be removed or commented out of the batch files.

MAKETABL Output Files

The filenames for the output of this program are supplied in the MAKEXX.INF files. The output of MAKETABL is the final table of attenuation coefficients, in the form illustrated in Figure 4 of the main report. The output files have been given names of the form FNAME = [m][a][λ].OUT. Here, [λ] is an integer (normally three or four digits) indicating at which wavelength the attenuation coefficients contained in the file have been calculated. [m] and [a] are one- or two-letter abbreviations of the atmosphere and aerosol models:

Table A-2. Naming Convention for MAKETABL Output Files.

FNAME = [m][a][λ].OUT	
[m] (atmosphere)	[a] (aerosol)
u = 1976 US Standard t = Tropical ms = Midlatitude Summer mw = Midlatitude Winter	r = Rural u = Urban m = Maritime d = Desert

Example: The file MSU1064.OUT lists the attenuation coefficients for λ = 1.064 μm, Midlatitude Summer atmosphere and Urban aerosol.

Files Transferred from the PC to the UNIX Stations

The attenuation table file, FNAME.OUT, is finally transferred to the Sun Sparcstations for use with LTAS. DOS and UNIX have a different format for the end-of-line characters in a file. The DOS format files contain an extra ^M character at the end of each line.

The final form of the attenuation coefficient tables, with this ^M character stripped from the DOS files, is stored under the name FNAME.DAT.

APPENDIX B: INPUT FILE FOR LNFL92

APPENDIX B: INPUT FILE FOR LNFL92

The program LNFL92.EXE is used to extract molecular line absorption data from the HITRAN92 data base. The program requires an input file TAPE5, an example of which is presented in Figure B-1.

```
10.6 um
 890.        1000.
1111111111111111111111111111111                              NLTE   NBLK1
```

Figure B-1. Sample LNFL92 input file TAPE5.

The first line of this file is a label line. The second line gives the wavenumber limits, in cm^{-1}, for which data should be extracted. These limits must be at least 25 cm^{-1} above and below the wavenumber of interest.

The last line indicates which molecules should be included when extracting the absorption data. There are 31 molecules in the HITRAN92 data base.

This particular input file is used to extract data for the λ = 10.6-μm calculations (wavenumber $k = 1/\lambda = 943.4\ cm^{-1}$). All 31 molecules are included in the extraction.

APPENDIX C: TEMPLATE FOR FASCODE INPUT FILE TAPE5

APPENDIX C: TEMPLATE FOR FASCODE INPUT FILE TAPE5

The following file, TEMPLATE.TP5, is a template for the FASCODE input file TAPE5. The TRANSMIT.BAT and TRANSDES.BAT batch files will copy this file to TAPE5 before using FASIN to modify it for the current calculations. TEMPLATE.TP5 will be included in the directory in which calculations are being made (i.e., TRANSMIT or TRANSDES executed).

For a full description of this input file, see the User Instructions for FASCOD3. This file is set up to calculate atmospheric optical depths for a given laser wavelength for ten values of slant range S_r. This input file is specifically set up for use with one of the four standard atmosphere models and one of the four aerosol models listed in Table 1 of the main report. Of course, this file can be altered to suit the needs of a specific calculation (e.g., calculate for a smaller number of slant paths for wavelengths for which the molecular absorption data file TAPE3 is unusually large).

Following the recommendations of the User Instructions, the CO_2 mixing ratio has been set to 360 ppmv, overriding the FASCODE default value of 330 ppmv (retained in the program for historical reasons).

```
$   Template Tape5, Sr = 1.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      1.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 2.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      2.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 3.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      3.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 4.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      4.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 5.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      5.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 6.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      6.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 7.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      7.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 8.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      8.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 9.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000      9.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
$   Template Tape5, Sr = 10.0 km
    1    1    1    1    0    0    0    0    0    1    0    1    0    0    0    0
 9398.500      0.000      0.000      0.000      0.000      0.000      0.000      0.000
    6    2    0    0    0   31    0           0.000      0.000      0.000    360.000
     0.000      0.152      0.000     10.000      0.000    0
     0.000      0.000      0.000      0.000      0.000
    1    0    0    0    0    0   23.500      0.000      0.000      0.000      0.000
%
```

Figure C-1. Listing of TEMPLATE.TP5.

APPENDIX D: FASIN

APPENDIX D: FASIN

The program FASIN.EXE is used to modify the FASCODE input file TAPE5. The new parameters are entered through a series of command line switches. This program is specifically designed to modify input files of the form of TEMPLATE.TP5, listed in Appendix C. To extend the system to include a user-defined atmosphere, this program will need to be modified, or another one written to handle such situations.

FASIN reads the file TAPE5, and writes the new file to TAPE5.NEW. This file must be copied over the old version of TAPE5 in order to use it with FASCODE. This protects against accidentally making unwanted changes to the input file.

Usage: `fasin /s=value </s=value ...>`

The available switches are listed in Table D-1.

Table D-1. Command Line Switches for FASIN.

Switch	Purpose
/m	input new atmosphere model (integer value)
/a	input new aerosol model (integer value)
/h1	input new starting altitude in ft
/h2	input new endpoint altitude in ft
/v	input new visibility in km
/w	input new wind speed in m/s
/l	input new wavelength in μm
/g	input new ground altitude in km
/c	input new CO_2 mixing ratio in ppmv

FASIN converts the altitudes from feet to kilometers, and calculates the wavenumber k in cm^{-1} from the wavelength before writing to TAPE5.NEW. Also, FASIN will insure that the slant range is greater than the altitude difference between the start and end points of the slant path (that is, it ensures that $S_r > h_2 - h_1$).

The starting point altitude is always set to h1 = 0.0 for purposes of making the attenuation coefficient tables.

Example: `fasin /a=1 /v=15.0 /l=0.532 /h2=15000.0`

This sets the aerosol to the Rural model (/a=1)
ground-level visibility to 15.0 km (/v=15.0)
wavelength to 0.532 μm (/l=0.532)
endpoint altitude to 15,000 ft (/h2=15000.0)
Note that there are no spaces around the "=" in the switches.

There are three files that make up the program FASIN. FASIN.CPP contains the main program. FASCARD.H and FASCARD.CPP define C++ structures representing the "cards" or lines that make up the input file TAPE5. These two files also declare and define overloaded C++ iostream insertion and extraction operators for use with the card structures.

Not all capabilities of FASIN are used in the system for making the attenuation coefficient tables. The ground altitude and the slant path initial altitude h_1 are always set to 0.0. The CO_2 mixing ratio is always set to 360 ppmv.

```
// fasin.cpp
//    Brian Lund, May 7, 1996
//    Alters the FASCODE input file tape5, using the "standard"
//    atmosphere, aerosol models
//
//    Usage:  fasin /s=value </s=value ...>
//    The switches are:
//          /m    inputs a new atmosphere model (integer value)
//          /a    inputs a new aerosol model (integer value)
//          /h1   inputs a new starting altitude in ft (float)
//          /h2   inputs a new endpoint altitude in ft (float)
//          /v    inputs a new visibility in km (float)
//          /w    inputs a new wind speed in m/s (float)
//          /l    inputs a new wavelength in um (float)
//          /g    inputs a new ground altitude in km (float)
//          /c    inputs a new CO2 mixing ratio (float)
//
//    The program reads the file tape5, and writes output to the
//    file tape5.new, which will need to be renamed or copied to
//    tape5 before running FASCODE.
//
//    Example:   fasin /a=1 /v=23.5 /l=1.064 /h2=500.0
//    sets the aerosol to the rural model (/a=1)
//             ground visibility to 23.5 km (/v=23.5)
//             wavelength to 1.064 um (/l=1.064)
//             endpoint altitude to 500 ft (/h2=500.0)
//

#include <iostream.h>
#include <fstream.h>
#include <strstream.h>
#include "fascard.h"
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

// editing flags
const unsigned MODEL_FLAG = 1;
const unsigned AEROSOL_FLAG = 2;
const unsigned H1_FLAG = 4;
const unsigned H2_FLAG = 8;
const unsigned VIS_FLAG = 16;        // visibility
const unsigned WIND_FLAG = 32;       // wind speed
const unsigned LAMBDA_FLAG = 64;     // wavelength
const unsigned GALT_FLAG = 128;      // ground altitude
const unsigned CO2_FLAG = 256;
const char START_FLAG = '$';         // indicated beginning of input
```

```
const char END_FLAG = '%';          // indicates end of tape5 input file

// useful constants
const float KMTOFT = 3280.8;        // convert kilometers to feet
const int MAXLEN = 81;              // file input buffer length
// structure for updated parameters
struct newinfo
      {
      int model, aerosol;
      float h1, h2, vis, wss, lambda, galt, co2;
      };

// input buffer array
char buffer[MAXLEN];

// function prototypes
unsigned parse(char *arg, newinfo& ni);

int main(int argc, char* argv[])

   Card1_2 card12 = { 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0 };
   Card1_3 card13 = { 9398.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
   Card3_1 card31 = { 6, 2, 0, 0, 1, 31, 0, 0.0, 0.0, 0.0, 360.0 };
   Card3_2 card32 = { 0.0, 0.152, 0.0, 1.0, 0.0, 0 };
   Card3_3a card33a = { 0.0, 0.0, 0.0, 0.0, 0.0 };
   Card4_1 card41 = { 1, 0, 0, 0, 0, 0, 23.5, 0.0, 0.0, 0.0, 0.0 };
         // The cards are initialized to some typical values
         //     1976 US std. atmos., rural aerosol, 23.5 km vis
         //          uses 31 molecules, h2 = 500 ft, Sr = 1 km
         //          wavelengh = 1064 nm
         // These values are never used, but are for reference
      newinfo update = { 0, 0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
         // This structure will hold the valued of the new parameters
         //    to be written to tape5.new
      unsigned uflag = 0;     // update flag
      float sroffset = 0.0;   // used for ensuring range > h2

      if (argc < 2)           // if no switches were passed to fasin
            {
            cout << endl << "ERROR: fasin requires at least one parameter!";
            cout << endl << endl;
            exit(1);
            }

      for(int i = 1; i < argc; I++)       // parse arguments, set member of
      uflag += parse(argv[i], update);    // update structure to new value
                                          // set bit of uflag

      ifstream infile("tape5", ios::in);  // Can't open tape5?
      if (!infile)
            {
            cout << endl << "Error opening tape5 for input" << endl;
            exit(1);
            }

      ofstream outfile("tape5.new", ios::out);  // Can't open tape5.new?
      if (!outfile)
      {
      cout << endl << "Error opening tape5.new for output" << endl;
      exit(1);
      }
```

```
      infile.getline(buffer, MAXLEN);      // read first line of tape5
      while( buffer[0] != END_FLAG )       // read until '%' encountered
            {
            while( buffer[0] != START_FLAG )// read until '&' encountered
            {
            outfile << buffer << endl;
            infile.getline(buffer, MAXLEN);
            } // end while search for start of input data
      outfile << buffer << endl;     // output line containing '$'

      infile >> card12 >> card13 >> card31 >> card32 >> card33a >> card41;

            // change parameters to new values
            if ( uflag & MODEL_FLAG ) card31.model = update.model;
            if ( uflag & AEROSOL_FLAG ) card41.ihaze = update.aerosol;
            if ( uflag & H1_FLAG ) card32.h1 = update.h1/KMTOFT;
            if ( uflag & H2_FLAG ) card32.h2 = update.h2/KMTOFT;
            if ( uflag & VIS_FLAG ) card41.vis = update.vis;
            if ( uflag & WIND_FLAG ) card41.wss = update.wss;
            if ( uflag & LAMBDA_FLAG ) card13.v1 = 10000./update.lambda;
            if ( uflag & GALT_FLAG ) card41.gndalt = update.galt;
            if ( uflag & CO2_FLAG ) card31.co2mx = update.co2;

            card32.range += sroffset;
            float dh = fabs(card32.h2 - card32.h1);
            if (card32.range < dh)
                  {
                  sroffset = ceil(dh) - card32.range;
                  card32.range += sroffset;
                  }

            outfile << card12 << card13 << card31 << card32 << card33a <<
card41;

            infile.ignore();
            infile.getline(buffer, MAXLEN);
            } // end while !END_FLAG

      infile.close();
      outfile << '%' << endl;
      outfile.close();

      cout << endl << "Output written to tape5.new." << endl;

      return 0;
      }  // end main()

unsigned parse(char *arg, newinfo& ni)  // parse switch arguments from
      {                                 // command line
      char sw[5];
      unsigned flagval = 0;
      istrstream ibuf(arg, strlen(arg));

      ibuf.get( sw, 5, '=');  // extract switch from argument string
      ibuf.ignore();          // skip '='

      if ( !strcmp(sw, "/m") )                        // update new info
      { ibuf >> ni.model;  flagval = MODEL_FLAG; } //  and set flag
      else if ( !strcmp(sw, "/a") )                  //  indicating which
      { ibuf >> ni.aerosol;  flagval = AEROSOL_FLAG; }  //  parameter has been
      else if ( !strcmp(sw, "/h1") )                     //  altered
```

```
{ ibuf >> ni.h1;  flagval = H1_FLAG; }
else if (!strcmp(sw, "/h2") )
{ ibuf >> ni.h2;  flagval = H2_FLAG; }
else if ( !strcmp(sw, "/v") )
{ ibuf >> ni.vis;  flagval = VIS_FLAG; }
else if ( !strcmp(sw, "/w") )
{ ibuf >> ni.wss;  flagval = WIND_FLAG; }
else if ( !strcmp(sw, "/l") )
{ ibuf >> ni.lambda;  flagval = LAMBDA_FLAG; }
else if ( !strcmp(sw, "/g") )
{ ibuf >> ni.galt;  flagval = GALT_FLAG; }
else if ( !strcmp(sw, "/c") )
{ ibuf >> ni.co2;  flagval = CO2_FLAG; }

return flagval;
}  // end parse()
```

```
// fascard.h
//          Brian Lund, May 7, 1996
//    Structure definitions and declarations for overloading iostream
//          insertion and extraction operators to use these structures, for
//          use in fasin.cpp
//
//          Currently set up only for use of standard FASCODE atmosphere and
//          aerosol models.

#include <iostream.h>

// Card structure definitions -
//          Each card is one line of a "standard" FASCODE input file TAPE5

// Card 1.1 is merely comment or label lines
// Card 1.2.1 not used for laser transmittance calculations
// Card 1.4 not used for laser transmittance calcs. (only optical depth
//   needed)
// Cards 2.x.y not needed when using standard models

struct Card1_2
      {
      int ihirac, ilblf4, icntnm, iaersl, iemit, iscan, ifiltr,
            iplot, itest, iatm, imrg, ilas, ims, ixsect, mpts, npts;
      };

struct Card1_3
      {
      float v1, v2, sample, dvset, alfalo, avmass, dptmin, dptfac;
      };

struct Card3_1
      {
      int model, itype, ibmax, nozero, noprnt, nmol, ipunch;
      float re, hspace, vbar, co2mx;
      };

struct Card3_2
      {
      float h1, h2, angle, range, beta;
      int len;
      };

struct Card3_3a
      {
      float avtrat, tdiff1, tdiff2, altd1, altd2;
      };

struct Card4_1
      {
      int ihaze, iseasn, ivulcn, icstl, icld, ivsa;
      float vis, wss, whh, rainrt, gndalt;
      };

// Function declarations - stream insertion and extraction operators
//          overloaded for use with the card structures.  The << operators
//          are set up to write the cars in the form specified by the
//          fortran formats used by FASCODE (see Fascode instructions)

ostream& operator <<(ostream& output, Card1_2& c);
ostream& operator <<(ostream& output, Card1_3& c);
```

```
ostream& operator <<(ostream& output, Card3_1& c);
ostream& operator <<(ostream& output, Card3_2& c);
ostream& operator <<(ostream& output, Card3_3a& c);
ostream& operator <<(ostream& output, Card4_1& c);

istream& operator >>(istream& input, Card1_2& c);
istream& operator >>(istream& input, Card1_3& c);
istream& operator >>(istream& input, Card3_1& c);
istream& operator >>(istream& input, Card3_2& c);
istream& operator >>(istream& input, Card3_3a& c);
istream& operator >>(istream& input, Card4_1& c);
```

```
// fascard.cpp
//          Brian Lund,  May 7, 1996
//          Stream insertion and extraction operators overloaded for use
//          with FASCODE input file modification program FASIN

#include <iostream.h>
#include <iomanip.h>
#include "fascard.h"

// Output stream (insertion) operators

ostream& operator <<(ostream& output, Card1_2& c)
      {
      output << setw(5) << c.ihirac
                  << setw(5) << c.ilblf4
                  << setw(5) << c.icntnm
                  << setw(5) << c.iaersl
                  << setw(5) << c.iemit
                  << setw(5) << c.iscan
                  << setw(5) << c.ifiltr
                  << setw(5) << c.iplot
                  << setw(5) << c.itest
                  << setw(5) << c.iatm
                  << setw(5) << c.imrg
                  << setw(5) << c.ilas
          << setw(5) << c.ims
                  << setw(5) << c.ixsect
                  << setw(5) << c.mpts
                  << setw(5) << c.npts << endl;

      return output;
      }

ostream& operator <<(ostream& output, Card1_3& c)
      {
      output.setf(ios::fixed);
      output.precision(3);
      output << setw(10) << c.v1
                  << setw(10) << c.v2
                  << setw(10) << c.sample
                  << setw(10) << c.dvset
                  << setw(10) << c.alfalo
                  << setw(10) << c.avmass
                  << setw(10) << c.dptmin
                  << setw(10) << c.dptfac << endl;

      return output;
      }

ostream& operator <<(ostream& output, Card3_1& c)
      {
      output << setw(5) << c.model
                  << setw(5) << c.itype
                  << setw(5) << c.ibmax
                  << setw(5) << c.nozero
                  << setw(5) << c.noprnt
                  << setw(5) << c.nmol
                  << setw(5) << c.ipunch;
      output.setf(ios::fixed);
      output.precision(3);
      output << setw(15) << c.re
```

```
                    << setw(10) << c.hspace
                    << setw(10) << c.vbar
                    << setw(10) << c.co2mx << endl;

        return output;
        }

ostream& operator <<(ostream& output, Card3_2& c)
        {
        output.setf(ios::fixed);
        output.precision(3);
        output << setw(10) << c.h1
                    << setw(10) << c.h2
                    << setw(10) << c.angle
                    << setw(10) << c.range
                    << setw(10) << c.beta;
        output.unsetf(ios::fixed);
        output << setw(5) << c.len << endl;

        return output;
        }

ostream& operator <<(ostream& output, Card3_3a& c)
        {
        output.setf(ios::fixed);
        output.precision(3);
        output << setw(10) << c.avtrat
                    << setw(10) << c.tdiff1
                    << setw(10) << c.tdiff2
                    << setw(10) << c.altd1
                    << setw(10) << c.altd2 << endl;

        return output;
        }

ostream& operator <<(ostream& output, Card4_1& c)
        {
        output << setw(5) << c.ihaze
                    << setw(5) << c.iseasn
                    << setw(5) << c.ivulcn
                    << setw(5) << c.icstl
                    << setw(5) << c.icld
                    << setw(5) << c.ivsa;
        output.setf(ios::fixed);
        output.precision(3);
        output << setw(10) << c.vis
                    << setw(10) << c.wss
                    << setw(10) << c.whh
                    << setw(10) << c.rainrt
                    << setw(10) << c.gndalt << endl;

        return output;
        }

// Input stream (extraction) operators

istream& operator >>(istream& input, Card1_2& c)
        {
        input >> c.ihirac >> c.ilblf4 >> c.icntnm >> c.iaersl >> c.iemit
                    >> c.iscan >> c.ifiltr >> c.iplot >> c.itest >> c.iatm
```

```
                    >> c.imrg >> c.ilas >> c.ims >> c.ixsect >> c.mpts >>
c.npts;

      return input;
      }

istream& operator >>(istream& input, Card1_3& c)
      {
      input >> c.v1 >> c.v2 >> c.sample >> c.dvset >> c.alfalo
                    >> c.avmass >> c.dptmin >> c.dptfac;

      return input;
      }

istream& operator >>(istream& input, Card3_1& c)
      {
      input >> c.model >> c.itype >> c.ibmax >> c.nozero >> c.noprnt
                    >> c.nmol >> c.ipunch >> c.re >> c.hspace >> c.vbar >>
c.co2mx;

      return input;
      }

istream& operator >>(istream& input, Card3_2& c)
      {
      input >> c.h1 >> c.h2 >> c.angle >> c.range >> c.beta >> c.len;

      return input;
      }

istream& operator >>(istream& input, Card3_3a& c)
      {
      input >> c.avtrat >> c.tdiff1 >> c.tdiff2 >> c.altd1 >> c.altd2;

      return input;
      }

istream& operator >>(istream& input, Card4_1& c)
      {
      input >> c.ihaze >> c.iseasn >> c.ivulcn >> c.icstl >> c.icld
                    >> c.ivsa >> c.vis >> c.wss >> c.whh >> c.rainrt >>
c.gndalt;

      return input;
      }
```

APPENDIX E: FASREAD

APPENDIX E: FASREAD

The program FASREAD is used to extract slant ranges and transmittances from the FASCODE output file TAPE6. It is tailored for laser transmittance (monochromatic) calculations in which FASCODE is used to calculate the optical depth of the atmosphere for the given wavelength and slant path. FASREAD will convert the optical depth τ to transmittance $T = \exp(-\tau)$ for output.

Usage: `fasread`

An example of the output of FASREAD is shown in Figure E-1. A FASCODE input file (of the form of TEMPLATE.TP5, see Appendix C) would have been set up to run calculations for the twelve values of S_r listed in the figure. The last line gives the negative of the altitude h for which the calculations were performed.

```
Sr    (km)       Trans.
      1.00       0.8950
      2.00       0.8013
      3.00       0.7172
      4.00       0.6413
      5.00       0.5749
      6.00       0.5138
      7.00       0.4601
      8.00       0.4131
      9.00       0.3698
      10.00      0.3303
      20.00      0.1076
      30.00      0.0349
-0.1520 km
```

Figure E-1. Example of FASREAD output.

This output can be piped into the program EFIT to produce an exponential fit of T versus S_r. EFIT will use the negative value in the last line as a flag indicating end of input.

```
// fasread.cpp
//          Brian Lund,  May 6, 1996
//          Reads FASCD3P output file TAPE6, extracting altitude (h2), slant
range
//          (Sr) and optical depth (tau).  Output is slant range and
transmittance
//          (T = exp(-tau)).  The last line of the output is negative of the
//          altitude (-h2 in km).  This output is in a form usable by EFIT to
//          produce exponential fits to T vs Sr.

#include <iostream.h>
#include <fstream.h>
#include <string.h>
#include <iomanip.h>
#include <math.h>

const char *PATH_FLAG = "CONTROL CARD 2.2";     // flag for path info
const char *TRANS_FLAG = "LASER OPTION  (AEROSOLS EFFECTS INCLUDED)";
                                                // flag for optical depth info
const int MAXLEN = 133;                    // maximum TAPE6 line length

void output(float r, float t);      // prototype - outputs sr, t

void main()
      {
      float h1, h2, angle, sr, k, tau, trans;
      int outflag = 0;
      char buffer[MAXLEN] = { "" };

      ifstream infile("tape6", ios::in);

      cout << setw(8) << "Sr (km)" << setw(8) << "Trans." << endl;
                                                                  // header
for output

      while (infile.peek() != EOF)        // loop until end of file
            {
            infile.getline(buffer, MAXLEN);

            if ( strstr(buffer, PATH_FLAG) != NULL )  // found path info
                  {
                  infile.ignore(MAXLEN, '=');         // Path info reproduced
from
                  infile >> h1;                      // TAPE5 input file
                  infile.ignore(MAXLEN, '=');
                  infile >> h2;
                  infile.ignore(MAXLEN, '=');
                  infile >> angle;
                  infile.ignore(MAXLEN, '=');
                  infile >> sr;
                  outflag++;                                          //
outflag = 1 -> path info read
                  }     // end if PATH_FLAG

            if ( strstr(buffer, TRANS_FLAG) != NULL ) // optical depths found
                  {
                  infile.getline(buffer, MAXLEN);                 // skip line
                  do {
                     infile.getline(buffer, MAXLEN);     // jump down to
wave no.
                     } while ( buffer[0] != '0' );       // of interest
```

```
                    infile >> k >> tau;                                  // k = wave
no., tau = opt. depth
                    trans = exp(-tau);
                    outflag++;                  // outflag = 2 -> both sr and tau
have been read
                    }     // end if TRANS_FLAG

              if ( outflag == 2 )               // we have both sr and tau
                    {
                    output( sr, trans );
                    outflag = 0;                              // reset flag to read
new data
                    }  // end if outflag

              }  // end while(infile)

        infile.close();
        cout << -h2 << " km" << endl;

        }   // end main()

void output(float r, float t)                   // function to output sr and t
        {
        cout.setf(ios::fixed);
        cout.precision(2);
        cout << setw(8) << r;
        cout.precision(4);
        cout << setw(8) << t;
        cout << endl;

        return;
    }
```

APPENDIX F: EFIT

APPENDIX F: EFIT

The program EFIT takes the output of FASREAD and produces an exponential fit of transmittance T versus slant range S_r. It uses the standard least squares method to produce the fit.

Usage: `fasread | efit`

or `efit < fname.dat`

where the file FNAME.DAT contains the output of FASREAD.

An example of the output of EFIT for a fit of the data listed in Figure E-1 is shown in Figure F-1. Note that the altitude has been converted to feet. This altitude is different from the value h = 500 ft originally input into FASCODE due to round-off errors in the conversions from feet to meters and back to feet again.

```
T(r) = T0*exp(-mu*r)
   Altitude = -498.682 ft
   T0 = 1.00572
   mu = 0.111853 km-1
   R^2 = 0.999982
```

Figure F-1. Sample output of EFIT.

The batch files TRANSMIT.BAT and TRANSDES.BAT append all the EFIT calculations into one file, which is then organized in a neater manner using MUTAB.

```
// efit.cpp
//          Brian Lund - May 6, 1996
//    Fits data to the form y = a*exp(bx)
//    Designed for use with MSREAD (MODTRAN) or FASREAD (FASCD3P)
//          Requires output of these programs can be piped into efit to
produce
//          a fit of transmittance vs slant range, T(Sr) = T0*exp(-u*Sr)
//
//       Usage:   fasread | efit >> output.dat
//
//                                fasread reads output of FASCD3P (T vs Sr) and
pipes to
//                                efit, which fits the data, with the resulting
fit appended
//                                to the file output.dat (normal output is to
screen)

#include <iostream.h>
#include <math.h>

const int MAXLEN = 80;          // input buffer length

int main()
      {
      char buffer[MAXLEN];
      int n = 0;                              // number of points
      float sumx = 0.0;
      float sumx2 = 0.0;
      float sumy = 0.0;
      float sumy2 = 0.0;
      float sumxy = 0.0;
      float r, t, h2;

      cin.getline(buffer, MAXLEN);  // skip label line

      do
            {
            cin >> r;
            if (r < 0.0)
                  {
                  h2 = 3280.8*r;                        // convert km to feet
                  break;
                  }
            float x = r;
            cin >> t;
            float y = log(t);
            sumx += x;
            sumx2 += x*x;
            sumy += y;
            sumy2 += y*y;
            sumxy += x*y;
            n++;
            } while ( r > 0.0);

//  fit coefficients
      float delta = n*sumx2 - sumx*sumx;
      float a = (sumx2*sumy - sumx*sumxy)/delta;
      float b = (n*sumxy - sumx*sumy)/delta;

// regression coeff. (goodness of fit)
      float r2 = b*(n*sumxy - sumx*sumy)/(n*sumy2 - sumy*sumy);
```

```
// put fit results in exponential form, t = t0*exp(-mu*r)
      float t0 = exp(a);
      float mu = -b;

      cout << "T(r) = T0*exp(-mu*r)" << endl;
      cout << "   Altitude = " << h2 << " ft" << endl;
      cout << "   T0 = " << t0 << endl;
      cout << "   mu = " << mu << " km-1" << endl;
      cout << "   R^2 = " << r2 << endl;

      return 0;
      }
```

APPENDIX G: MUTAB

APPENDIX G: MUTAB

The program MUTAB organizes the output of the EFIT calculations into a somewhat neater tabular form. It corrects for the round-off errors of the altitudes demonstrated in Figure F-1. A sample output is shown in Figure G-1.

Usage: `mutab fname.ext`

where FNAME.EXT is the name of the file containing the results of the EFIT calculations.

ALT (FT)	TO	MU (KM-1)	R^2
1000.0	1.00098	0.2011920	0.999993
2000.0	1.00048	0.2008350	0.999999
3000.0	1.00055	0.2009960	1.000000
4000.0	1.00058	0.1939570	0.999999
5000.0	1.00079	0.1743530	1.000000
6000.0	1.00073	0.1539590	0.999999
7000.0	1.00080	0.1361730	0.999999
8000.0	1.00101	0.1219760	0.999998
9000.0	1.00087	0.1099840	0.999999
10000.0	1.00100	0.1009720	1.000000
11000.0	1.00102	0.0928439	1.000000
12000.0	1.00102	0.0858867	1.000000
13000.0	1.00103	0.0799169	1.000000
14000.0	1.00112	0.0746791	0.999999
15000.0	1.00096	0.0700587	0.999998
16000.0	1.00099	0.0659670	1.000000
17000.0	1.00133	0.0623411	0.999999
18000.0	1.00123	0.0590791	1.000000
19000.0	1.00089	0.0561226	1.000000
20000.0	1.00131	0.0535039	1.000000

Figure G-1. Sample of Output from MUTAB.

```
// mutab.cpp
//          Brian Lund, May 6, 1996
//    Assembles table of exponential fits from EFIT output
//          which has been written to the file infile
//    Usage:  mutab infile > outfile
//    reads the file infile containing results of fits from EFIT
//                and organizes them in the file outfile (normal output is
//                   to screen)

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <strstrea.h>
#include <math.h>
#include <string.h>

int main(int argc, char* argv[])
      {
      char buffer[80];
      char str1[20], str2[5];
      float h, t0, mu, r2, th;

      ifstream infile(argv[1], ios::in);     // file read from command line

      cout << setw(10) << "ALT (FT)"         // write table column labels
             << setw(10) << "T0"
             << setw(15) << "MU (KM-1)"
             << setw(10) << "R^2" << endl;

      while(infile.peek() != EOF)            // read until end of file on
infile
            {
            for(int i=0; i<5; i++)
                  {
                  infile.getline(buffer, 80);      // skip label line
                  istrstream inBuf(buffer, strlen(buffer));
                  switch(i)
                        {
                        case 0 : break;
                        case 1 : inBuf >> str1 >> str2 >> th; break;
                        case 2 : inBuf >> str1 >> str2 >> t0; break;
                        case 3 : inBuf >> str1 >> str2 >> mu; break;
                        case 4 : inBuf >> str1 >> str2 >> r2; break;
                        }  // end switch
                  }   // end for
            h = floor(-th/250.0 + 0.5)*250.0;   // corrects alt. in ft
            cout.setf(ios::fixed);              // for roundoff errors
            cout.precision(1);                  // should be an integer
            cout << setw(10) << h;              // multiple of 250 ft.
            cout.precision(5);
            cout << setw(10) << t0;
            cout.precision(7);
            cout << setw(15) << mu;
            cout.precision(6);
            cout << setw(10) << r2 << endl;
            }  // end while()

      return 0;
      }
```

APPENDIX H: MAKETABL

APPENDIX H: MAKETABL

The program MAKETABL assembles the final table of attenuation coefficients. Figure 4 of the main report is an example of the output of MAKETABL. This program is written specifically to create columns of attenuation coefficients for visibility V = 5.0 km, 8.0 km, 15.0 km, 23.5 km, and 40.0 km, and for altitudes from 1000 ft to 20,000 ft.

The program is controlled through an input file. An example of an input file used to assemble the four tables (one for each aerosol model) for the 1976 US Standard atmosphere for λ = 0.532 μm is shown in Figure H-1. Note that one input file can be used to assemble several tables.

```
ur532.out
1976 US Standard, Rural, 0.532 um
usrur5.tab
usrur8.tab
usrur15.tab
usrur235.tab
usrur40.tab
um532.out
1976 US Standard, Maritime, 0.532 um
usmar5.tab
usmar8.tab
usmar15.tab
usmar235.tab
usmar40.tab
uu532.out
1976 US Standard, Urban, 0.532 um
usurb5.tab
usurb8.tab
usurb15.tab
usurb235.tab
usurb40.tab
ud532.out
1976 US Standard, Desert, 0.532 um
usdes5.tab
usdes8.tab
usdes15.tab
usdes235.tab
usdes40.tab
```

Figure H-1. MAKEUS.INF, used by MAKETABL to Assemble Four Tables for 1976 US Standard Atmosphere for λ = 0.532 μm.

The first seven lines of this file control the assembly of the attenuation coefficient table for the Rural aerosol. The first line gives the final output filename; the tables will be written to UR532.OUT. The second line is a label that will be copied to UR532.OUT. The next five lines give the names of the MUTAB output files containing the results for V = 5.0 km, 8.0 km, 15.0 km, 23.5 km, and 40.0 km. These files must be in the order shown in order for the final tables to make sense.

Usage: `maketabl makexx.inf`

where MAKEXX.INF is the file controlling the assembly of the attenuation tables.

Four standard input files have been created, one for each of the four atmosphere models: MAKEUS.INF (1976 US Standard), MAKETR.INF (Tropical), MAKEMS.INF (Midlatitude Summer), and MAKEMW.INF (Midlatitude Winter). These files should be edited to reflect the wavelength for which attenuation coefficient tables are being created.

If there is a need to extend the tables to different altitudes or visibility ranges, this program will have to be modified.

```
// maketabl.cpp
//          Brian Lund, May 8, 1996
//                                   Updated May 11, 1996
//
//          Usage:   maketabl fname.ext
//          Assembles final attenuation coefficient tables from FASCODE
//          calculations.  Reads in files from MUTAB output, extracting
altitude
//          and attenuation coefficients.  Process is controlled by the input
//          file fname.exe.
//
//          Form for fname.ext:
//                line 1:  Output file name (file name for final tables)
//                line 2:  Label line (copied to output file)
//                line 3:     .TAB file for V = 5 km
//                line 4:  .TAB file for V = 8 km
//                line 5:  .TAB file for V = 15 km
//                line 6:  .TAB file for V = 23.5 km
//                line 7:  .TAB file for V = 40 km
//          Repeat lines 1-7 as many times as desired

#include <iostream.h>
#include <fstream.h>
#include <strstrea.h>
#include <iomanip.h>
#include <string.h>
#include <stdlib.h>

const int RECLEN = 81;          // max. line length for input files
const int NVIS = 5;          // # vis. ranges, currently 5 (see above)
const int NALT = 20;         // # altitudes, currently 20 (1000 to 20,000 ft)

int main(int argc, char *argv[])
      {
      float alt[NALT], mu[NVIS][NALT];
      float t0, r2;
      int i, j;
      char buffer[RECLEN];
      char outfilnam[30];
      char infilnam[30];

      ifstream infil;
      ofstream outfil;
```

```
      if (argc != 2)      // input file must be entered on command line
            {
            cout << endl << "Usage:  maketabl fname.ext." << endl;
            exit(0);
            }

      ifstream listfil(argv[1], ios::in);  // Problem opening input file?
      if (!listfil)
            {
            cout << endl << "Error opening " << argv[1] << " for read!" <<
endl;
            exit(0);
            }

      while (listfil.peek() != EOF)
            {
            listfil.getline(buffer, RECLEN);                        // read
output file name
            strcpy(outfilnam, buffer);
            outfil.open(outfilnam, ios::out);
            if (!outfil)                                  // exit on file i/o
error
                  {
                  cout << endl << "Error opening " << outfilnam << " for
output!"
                        << endl;
                  listfil.close();
                  exit(0);
                  }

            listfil.getline(buffer, RECLEN);                        // read
label line
            outfil << buffer << endl;                     //   then output

            outfil << setw(8) << "Alt." << "   Attenuation Coefficients (km-
1)"
                        << endl;                          // table header lines
            outfil << setw(8) << "(ft)"
                        << setw(12) << "V=5.0km"
                        << setw(12) << "V=8.0km"
                        << setw(12) << "V=15.0km"
                        << setw(12) << "V=23.5km"
                        << setw(12) << "V=40.0km" << endl;

            for( i=0; i<NVIS; i++)        // loop over the NVIS input files
                  {                          // to read in data
                  listfil.getline(buffer, RECLEN);
                  strcpy(infilnam,buffer);
                  infil.open(infilnam, ios::in);
                  if (!infil)
                        {
                        cout << endl << "Error opening " << infil << " for
input!" << endl;
                        exit(0);
                        }
                  infil.getline(buffer, RECLEN);            // skip column
headings
                  for( j=0; j<NALT; j++)                          // read in
for NALT altitudes
                        infil >> alt[j] >> t0 >> mu[i][j] >> r2;
                  infil.close();
```

```
                    }

            for(j=0; j<NALT; j++)                                   // write
table to output file
                    {                                       // 1st column = altitude
                    outfil.setf(ios::fixed);                // 2nd col = att. coef
for V=5 km
                    outfil.precision(1);                    // 3rd col =  " for V =
8 km
                    outfil << setw(8) << alt[j];                // 4th col =  "
for V = 15 km
                    outfil.unsetf(ios::fixed);                      // 5th col =
" for V = 23.5 km
                    outfil.setf(ios::scientific);               // 6th col =  "
for V = 40 km
                    outfil.precision(2);
                    outfil << setw(12) << mu[0][j]
                                << setw(12) << mu[1][j]
                                << setw(12) << mu[2][j]
                                << setw(12) << mu[3][j]
                                << setw(12) << mu[4][j]
                                << endl;
                    }
            outfil.close();         // close output (table) file when finished

//          listfil.ignore();       // skip EOL

      };    // end while listfil

      return 0;
      }
```

APPENDIX I: TRANSMIT.BAT

APPENDIX I: TRANSMIT.BAT

This batch file controls the calculations necessary to produce one column of the attenuation coefficient tables (e.g., Figure 4 of the main report). It is tailored for use with the Rural, Urban, and Maritime aerosol models.

Usage: TRANSMIT m a v λ FNAME

where
- m = atmosphere model identifier (see FASCODE instructions)
- a = aerosol model identifier
- v = ground level visibility (in km)
- λ = wavelength of laser (in μm)

The results of the calculations will be stored in FNAME.TAB. This file is the output of the program MUTAB, called at the end of the batch file. An intermediate file FNAME.DAT will be used to store the result of EFIT exponential fits.

Example: TRANSMIT 1 4 8.0 0.532 TRMAR532

This will calculate the atmospheric attenuation coefficients from altitudes h = 1000 ft to h = 20,000 ft in 1000-ft increments, using the Tropical atmosphere (m = 1), Maritime aerosol (a = 4), and a visibility of V = 8.0 km for a laser emitting light with the wavelength λ = 0.532 μm. The results will be written to the file TRMAR532.TAB.

Before running TRANSMIT, make sure that the file TAPE3 contains a portion of HITRAN92 which includes the wavenumber (k = 1/λ) of interest. Also, the file TEMPLATE.TP5 is necessary for the successful execution of TRANSMIT.

```
@echo off
rem transmit.bat
rem
rem     Brian Lund, May 7, 1996
rem
rem     Batch file to create atmospheric attenuation table using FASCODE
rem     for a given (standard) atmosphere model, aerosol model, visibility
rem     and wavelength.  The user must be sure that the file TAPE3 contains
rem     the proper molecular line data for the wavelength of interest.
rem
rem     Usage:  transmit m a v l fname
rem     where
rem             m = atmospheric model identifier (see FASCODE instructions)
rem             a = aerosol model identifier
rem             v = ground level visibility (in km)
rem             l = wavelength (in um)
rem     and the results of the calculations will be stored in the file
rem     fname.tab.  An intermediate file fname.dat will be used to
rem     temporarily store results
rem
rem     Example:  transmit 6 1 15.0 1.064 usrur15
rem     This will calculate atmospheric transmittance using the 1976 US
rem     standard atmosphere (6), the rural aerosol model (1), visiblity
```

```
rem     v = 15.0 km, for a wavelength of 1.064 um.  The results will be
rem     stored in usrur15.tab.
rem
copy template.tp5 tape5
fasin /m=%1 /a=%2 /v=%3 /h2=1000.0 /l=%4
copy tape5.new tape5
fascd3p
fasread | efit > %5.dat
echo 1000 ft
fasin /h2=2000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 2000 ft
fasin /h2=3000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 3000 ft
fasin /h2=4000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 4000 ft
fasin /h2=5000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 5000 ft
fasin /h2=6000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 6000 ft
fasin /h2=7000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 7000 ft
fasin /h2=8000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 8000 ft
fasin /h2=9000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 9000 ft
fasin /h2=10000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 10000 ft
fasin /h2=11000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 11000 ft
fasin /h2=12000.0
copy tape5.new tape5
```

```
fascd3p
fasread | efit >> %5.dat
echo 12000 ft
fasin /h2=13000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 13000 ft
fasin /h2=14000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 14000 ft
fasin /h2=15000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 15000 ft
fasin /h2=16000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 16000 ft
fasin /h2=17000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 17000 ft
fasin /h2=18000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 18000 ft
fasin /h2=19000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 19000 ft
fasin /h2=20000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %5.dat
echo 20000 ft
rem
rem now assemble final table of fits
mutab %5.dat > %5.tab
rem
echo model = %1
echo aerosol = %2
echo visibility = %3
echo wavelength = %4
echo fname = %5
```

APPENDIX J: TRANSDES.BAT

APPENDIX J: TRANSDES.BAT

This batch file is very similar to TRANSMIT.BAT. It controls the calculation of the attenuation coefficients for the Desert aerosol model, in which the visibility is determined by the current wind speed. Like TRANSMIT, TRANSDES will calculate the attenuation coefficients for one column of the attenuation tables.

Usage: TRANSDES m w λ FNAME

where m = atmosphere model identifier (see FASCODE instructions)
w = wind speed (in m/s)
λ = wavelength (in μm)

The results will be stored in FNAME.TAB, a MUTAB output file. An intermediate file FNAME.DAT is used to store the result of EFIT exponential fits.
Example: TRANSDES 6 21.5 1.064 USDES215

This will calculate the attenuation coefficients using the Desert aerosol, the 1976 US Standard atmosphere, and a wind speed of w = 21.5 m/s for light of wavelength λ = 1.064 μm. The results will be saved in the file USDES215.TAB.

Before running TRANSDES, make sure that the file TAPE3 includes a portion of HITRAN92 containing molecular absorption data for the wavenumber (k = 1/λ) of interest. The file TEMPLATE.TP5 is needed to run TRANSDES.

```
@echo off
rem transdes.bat
rem
rem     Brian Lund, May 10, 1996
rem
rem     Batch file to create atmospheric attenuation table using FASCODE
rem     for a given (standard) atmosphere model, visibility and wavelength,
rem     and desert aerosol model.  The user must be sure that the file TAPE3
rem     contains the proper molecular line data for the wavelength of
rem     interest.
rem
rem     Usage:  transmit m w l fname
rem     where
rem             m = atmospheric model identifier (see FASCODE instructions)
rem             w = wind speed (in m/s)
rem             l = wavelength (in um)
rem     and the results of the calculations will be stored in the file
rem     fname.tab.  An intermediate file fname.dat will be used to
rem     temporarily store results
rem
rem     Example:  transmit 6 15.0 1.064 usdes15
rem     This will calculate atmospheric transmittance using the 1976 US
rem     standard atmosphere (6), desert aerosol,  wind speed
rem     w = 15.0 km, for a wavelength of 1.064 um.  The results will be
rem     stored in usdes15.tab.
rem
```

```
copy template.tp5 tape5
fasin /m=%1 /a=10 /w=%2 /h2=1000.0 /l=%3 /v=0.0
copy tape5.new tape5
fascd3p
fasread | efit > %4.dat
echo 1000 ft
fasin /h2=2000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 2000 ft
fasin /h2=3000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 3000 ft
fasin /h2=4000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 4000 ft
fasin /h2=5000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 5000 ft
fasin /h2=6000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 6000 ft
fasin /h2=7000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 7000 ft
fasin /h2=8000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 8000 ft
fasin /h2=9000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 9000 ft
fasin /h2=10000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 10000 ft
fasin /h2=11000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 11000 ft
fasin /h2=12000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 12000 ft
```

```
fasin /h2=13000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 13000 ft
fasin /h2=14000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 14000 ft
fasin /h2=15000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 15000 ft
fasin /h2=16000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 16000 ft
fasin /h2=17000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 17000 ft
fasin /h2=18000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 18000 ft
fasin /h2=19000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 19000 ft
fasin /h2=20000.0
copy tape5.new tape5
fascd3p
fasread | efit >> %4.dat
echo 20000 ft
rem
rem now assemble final table of fits
mutab %4.dat > %4.tab
rem
echo model = %1
echo wind = %2
echo wavelength = %3
echo fname = %4
```

APPENDIX K: GOXX.BAT

APPENDIX K: GOXX.BAT

The four batch files, GOUS.BAT, GOTR.BAT, GOMS.BAT, and GOMW.BAT, control the assembly of the attenuation coefficient tables for each of the four atmosphere models of interest. Figure K-1 is a listing of GOMW.BAT, which is used to assemble tables for the Midlatitude Winter atmosphere model. It is presently set up to calculate tables for $\lambda = 0.850$ μm. The other three batch files differ only in the atmosphere model used in the calls to TRANSMIT and TRANSDES.

```
rem  Midlatitude Winter
rem
rem  Rural Aerosol
call transmit 3 1 40.0 0.850 mwrur40
call transmit 3 1 23.5 0.850 mwrur235
call transmit 3 1 15.0 0.850 mwrur15
call transmit 3 1 8.0 0.850 mwrur8
call transmit 3 1 5.0 0.850 mwrur5
rem  Maritime Aerosol
call transmit 3 4 40.0 0.850 mwmar40
call transmit 3 4 23.5 0.850 mwmar235
call transmit 3 4 15.0 0.850 mwmar15
call transmit 3 4 8.0 0.850 mwmar8
call transmit 3 4 5.0 0.850 mwmar5
rem  Urban Aerosol
call transmit 3 5 40.0 0.850 mwurb40
call transmit 3 5 23.5 0.850 mwurb235
call transmit 3 5 15.0 0.850 mwurb15
call transmit 3 5 8.0 0.850 mwurb8
call transmit 3 5 5.0 0.850 mwurb5
rem  Maritime Aerosol
call transdes 3 12.97 0.850 mwdes40
call transdes 3 17.21 0.850 mwdes235
call transdes 3 20.42 0.850 mwdes15
call transdes 3 23.45 0.850 mwdes8
call transdes 3 25.68 0.850 mwdes5
rem
rem  Make tables
maketabl makemw.inf
del *.dat
del *.tab
```

Figure K-1. Listing of GOMW.BAT for $\lambda = 0.850$ μm.

Before running these batch files, the wavelength parameter in the calls to TRANSMIT and TRANSDES must be changed to the value for which attenuation coefficients are to be calculated. The MAKEXX.INF files will also need to be edited to reflect this wavelength. In addition, the file TAPE3 must contain the molecular absorption data for the wavelength of interest.

The last two lines in Figure K-1 are used to delete the files used to contain intermediate results from the outputs of EFIT and MUTAB. Note that these lines are especially brutal; make

sure no files with the extensions .DAT or .TAB exist in the directory in which these batch files are being executed.

One last batch file, GOALL.BAT, was created to execute the four GOXX.BAT batch files to create all sixteen attenuation tables for a given wavelength. A listing of GOALL.BAT is shown in Figure K-2.

```
rem call gous
call gotr
call goms
call gomw
```

Figure K-2. Listing of GOALL.BAT.